ATK tähtitieteessä

Osa 3 - IDL proseduurit ja rakenteet

25. huhtikuuta 2017

IDL - proseduurit

- Viimeksi käsiteltiin IDL:n interaktiivista käyttöä, mutta tämä on hyvin kömpelöä monimutkaisempia asioita tehtäessä.
- IDL:llä on mahdollista tehdä ns. proseduuri-tiedostoja, jotka voidaan ajaa IDL-komentoriviltä. Tiedostot suoritetaan rivi kerrallaan, ja syntaksi on käytännössä sama kuin suoraan komentotilassa.
- Proseduureja on kolmea eri tyyppiä:
  - Pääohjelmat.
  - Aliohjelmat.
  - Funktiot.
- Proseduurin kutsuminen ja ajaminen riippuu sen tyypistä.
- Tiedostomuoto on muotoa **.pro**.
- Jokainen proseduuri päättyy riviin jossa lukee vain **END**.

IDL - proseduurit

- Hyvä ohjelmien kehitystapa on pitää editoitava ohjelma taustalla esimerkiksi Emacs-ikkunassa, tallennetaan muutokset ja katsotaan komentotilassa ohjelma ajamalla mitä muutokset ovat saaneet aikaan.
- Proseduuritiedostossa rivejä voi kommentoida pois, tai vastaavasti kommentteja voi lisätä omiksi muistiinpainoiksi lisäämällä rivin alkuun merkin **;**. Tällöin ohjelmaa ajaessa tätä riviä ei huomioida.
- Perusohjeena voidaan sanoa, että jos tarpeesi ovat taskulaskinkäyttöä monimutkaisempia on käytännöllisempää tehdä asiasta proseduuri.

IDL - pääohjelmat

- Pääohjelma-tyyppinen proseduuri käyttää samaa työtilaa kuin interaktiivinen käyttö. Toisin sanoen, ohjelman suorituksen jälkeen siinä määritellyt muuttujat ja parametrit jäävät aktiiviseksi komentotilaan.
- Yksinkertaisimmillaan pääohjelma on muotoa:

```
print,'Hello world!'
end
```

- Ohjelma ajetaan komennolla **.run tiedostonimi**. Esimerkiksi:

```
IDL> .run yksinkertainen.pro
% Compiled module: $MAIN$.
Hello world!
```

- Jos tiedostonimessä muistetaan käyttää .pro päätettä, voidaan se jättää komentotilassa huomiotta:

```
IDL> .run yksinkertainen
% Compiled module: $MAIN$.
Hello world!
```

IDL - pääohjelmat

- Ohjelman sisällä voidaan viitata muuttujaan eri kohdassa ohjelmaa, ja sen arvoa voidaan muuttaa ohjelman edetessä.
- Pääohjelmat ovat erittäin käytännöllisiä kun tarvitaan ohjelmia joissa on paljon erilaisia määrittelyitä ja halutaan tallentaa ne myöhempää käyttöä varten.
- Ohjelmiin voi sisällyttää erilaisia silmukkarakenteita (käydään läpi hieman myöhemmin).
- Pääohjelmien ongelmien etsiminen on myös helppoa. Jos suoritus pysähtyy virheeseen voi muuttujien arvoja katsoa komentotilassa.

IDL - aliohjelmat

- Aliohjelmilla on oma muuttuja-alueensa. Eli käytetyt muuttujat eivät siirry automaattisesti komentotilaan.
- Muuttujat ja niiden arvot voidaan kuitenkin siirtää komentotilaan (tai komentotilasta aliohjelmaan) ohjelman kutsussa määriteltyjen argumenttien avulla, jos näin halutaan.
- Aliohjelma-proseduuri määritellään sen alussa olevalla rivillä:

```
PRO aliohjelman_nimi, arg1, arg2, ..., keyword1=keyword1, keyword2=keyword2, ...
```

- Jossa kerrotaan aliohjelman nimi, ja mahdolliset argumentit sekä avainsanat (keyword). Aliohjelma myös päättyy riviin jossa lukee end.
- Aliohjelman nimen ei tarvitse olla sama kuin tiedostonimen, ja yhdessä tiedostossa voidaan määritellä useita eri aliohjelmia.

IDL - aliohjelmat

- Tässä esimerkki hyvin yksinkertaisesta aliohjelmasta:

```
pro aliohjelman_nimi,x,kappaletta
  ; määritellään aliohjelmassa x,
  ; ja käytetään toista argumenttia
  ; sen luomiseen
  x = findgen(kappaletta)
end
```

- Aliohjelma **käännetään** komennolla:

```
IDL> .run aliohjelma.pro
```

- Tämä ei kuitenkaan suorita sitä. **Suoritus** tapahtuu kirjoittamalla aliohjelman nimi ja argumentit pilkulla eroteltuna. Esimerkiksi ylläolevaa ohjelmaa voi kutsua komentotilasta tai toisesta ohjelmasta:

```
IDL> aliohjelman_nimi,uusimuuttuja,5
```

- Koska määrittelimme aliohjelman tallentamaan muuttujaan x taulukon, voimme aliohjelman suorituksen jälkeen katsoa sen arvoja:

```
IDL> print,uusimuuttuja
      0.00000 1.00000 2.00000 3.00000 4.00000
```


IDL - aliohjelmat

- On syytä muistaa muutama asia aliohjelmista:
  - Jos aliohjelmaa muuttaa, on muistettava kääntää se ennen uudelleen suorittamista. Muutoin muutokset eivät ole voimassa.
  - Mikäli aliohjelma päättyy virheeseen eivät varsinaisen työtilan muuttujien arvot yms. ole käytettävissä. Takaisin perustilaan pääsee antamalla komennon **retall**.
  - Aliohjelmaan syötettävät argumentit voivat olla muuttujia tai vakioita. Aliohjelmasta ulos saatavat tietojen (output-parametrien) on oltava muuttujia. Suoritettaessa annettavan argumentin nimen ei tarvitse olla sama kuin aliohjelman sisällä oleva nimi. Argumenttien järjestyksellä kutsussa on väliä.
  - Keywordien käyttöä kutsussa etsitään aliohjelmassa käyttämällä *keyword_set(keyword)*-rakennetta. Sen käytöstä lisää manuaalissa.

IDL - funktiot

- Funktio-rakenteen suurin ero aliohjelmaan on, että funktiot voivat palauttaa vain yhden muuttujan.
- Funktio määritellään sen alussa olevalla rivillä:

```
FUNCTION funktion_nimi, arg1, arg2, ..., keyword1=keyword1, keyword2=keyword2, ..
```

- Palautettava muuttuja määritellään funktion lopussa *return*-lauseella.
- Aiempi aliohjelma funktioksi muutettuna:

```
function funktion_nimi,kappaletta
  x = findgen(kappaletta)
  return,x
end
```

- Funktio käännetään kuten pää- ja aliohjelmat komennolla:

```
IDL> .run funktio.pro
```

- Funktiota kutsutaan komennolla:

```
muuttuja=funktion_nimi(arg1, arg2, ..., keyword1=keyword1, ...)
```


IDL - funktiot

- Esimerkiksi äskeinen yksinkertainen esimerkkiohjelma:

```
IDL> .run funktio.pro
IDL> uusimuuttuja=funktion_nimi(5)
IDL> print,uusimuuttuja
      0.00000 1.00000 2.00000 3.00000 4.00000
```

- Aliohjelmien yhteydessä mainitut muistettavat erityishuomioit pätevät myös funktioihin.

IDL - komentorakenteet

- Proseduureissa voidaan käyttää muun muassa seuraavia rakenteita:
  - FOR-silmukat,
  - IF-rakenteet,
  - WHILE-rakenne,
  - STOP-proseduuri.

IDL - FOR-silmukat

- FOR-rakenteella voidaan tehdä silmukoita, joissa haluttuja toimenpiteitä toistetaan.
- Esimerkiksi jos haluttu toimenpide on yksittäinen komento.
  - tulostaa luvut 0, 1, . . . , 10.
    `FOR i=0,10 DO print,i`
  - tulostaa luvut 0, 2, 4, 6, 8, 10.
    `FOR i=0,10,2 DO print,i`
- Esimerkeistä nähdään rakenteen tärkeät elementit:
  - `FOR` – aloittaa silmukan määrittelyn.
  - Silmukassa aktiivinen muuttuja jonka arvoja muutetaan. Ylläolevissa esimerkeissä muuttuja i.
  - Aktiivisen muuttujan arvoväli. Kaksi ensimmäistä pilkulla erotettua numeroa muuttujan esittelyn (`i=`) jälkeen.
  - Aktiivisen muuttujan arvon muutos, kolmas pilkulla erotettu numero. Ei pakollinen, vakiolisäys on $+1$.
  - `DO` – aloittaa itse silmukan.

IDL - FOR-silmukat

- Jos silmukassa suoritetaan useita komentoja käytetään seuraavaa rakennetta:

```
j=fltarr(10)
FOR i=0,9 DO BEGIN
  j(i)=i*i
  print,i,j(i)
ENDFOR
```

- Nyt `BEGIN` ilmoittaa että seuraa useita lauseita.
- `ENDFOR` päättää silmukan.
- Esimerkissä on myös rakenne, jossa taulukkon arvoja muutetaan silmukassa käytettävän muuttujan avulla. Tämä on hyvin yleistä.
  - On muistettava että IDL:ssä taulukon ensimmäisen alkion indeksi on 0, vastaavasti viimeisen on siis taulukon koko -1.

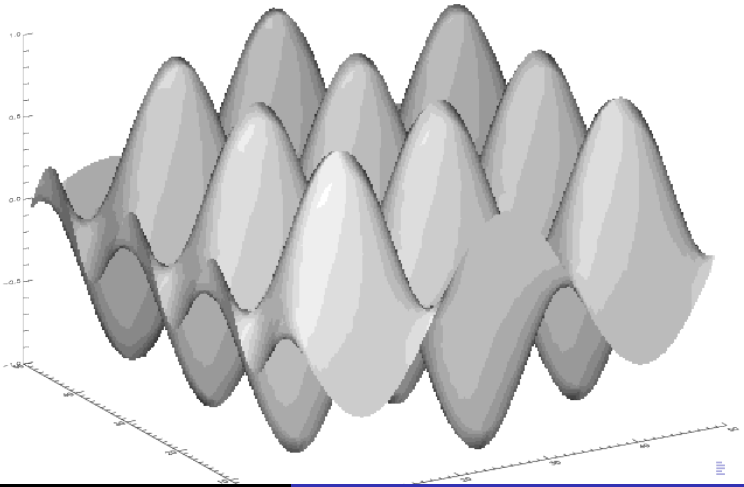
IDL - FOR-silmukat

- Tässä hieman monimutkaisempi esimerkki, jota voitte kokeilla myös harjoituksissa:

```
;taulukkomäärittelyt
x=fltarr(50,50) & y=x & z=x
;x- ja y-juoksevat numerot
FOR i=0,49 DO BEGIN
  FOR j=0,49 DO BEGIN
    x(i,j)=float(i)
    y(i,j)=float(j)
  ENDFOR
ENDFOR
;arvot välillä [-5,5]
x=x/5.-5.
y=y/5.-5.
;mielivaltainen z-pinta
z=sin(2*y)*cos(x)
;piirretään varjostettu pinta
shade_surf,z
END
```


IDL - FOR-silmukat

- Joka tuottaa seuraavaa:



IDL - IF-rakenne

- IF-rakenteella voidaan suorittaa komento, jos määritelty ehto täyttyy.
- Esimerkiksi jos haluttu toimenpide on yksittäinen komento.
  - Jos $x \neq 2$ tulostetaan tämä:

```
IF x NE 2 THEN print,'x ei ole 2'
```

  - Sulkuja voidaan myös käyttää selventämään koodia:

```
IF(x NE 2) THEN print,'x ei ole 2'
```

  - Lisää mahdollisia rakenteita:

```
IF(x EQ 2) THEN print,'x=2'
IF(x LE 2) THEN print,'x<=2'
IF(x GT 2 or LT 0) THEN print,'x>2 tai x<0'
IF(x GT 2 or LT 10) THEN print,'2<x<10'
IF(x GT 2 AND x NE 10) THEN print,'x>2 ja x!=10'
```

IDL - IF-rakenne

- Rakenne jos halutaan suorittaa useampia komentoja ehdon täyttyessä on seuraava:

```
IF(x EQ 0) THEN BEGIN
  print,'x=0'
  y=2.
ENDIF
```

- Nyt `THEN BEGIN` ilmoittaa että ehdon täyttyessä seuraa useita lauseita.
- `ENDIF` päättää komennot.
- Voidaan myös lisätä `ELSE`-rakenne joka suoritetaan jos ehto ei täyty:

```
IF(x EQ 0) THEN BEGIN
  print,'x=0'
  y=2.
ENDIF ELSE BEGIN
  print,'x!=0'
  y=1.
ENDELSE
```


IDL - WHILE-rakenne

- WHILE DO rakenteella suoritetaan toimintoa kunnes annettu ehto ei ole enää voimassa. Esimerkiksi:

```
IDL> x=0.
IDL> WHILE x LT 10. DO x=x+1.
IDL> print,x
      10.0000
```

- Taas jos halutaan suorittaa useita komentoja kunnes ehto ei ole enää voimassa:

```
x=0.
WHILE(x lt 10.) DO BEGIN
  x=x+1.
  print,x
ENDWHILE
```


IDL - STOP-proseduuri

- STOP-proseduurilla voidaan pysäyttää ohjelma suoritus halutussa kohdassa ja palata takaisin komentotilaan. Esimerkiksi:

```
stop,x,y
```

- Jolloin ohjelma pysähtyy ja printtaa x- ja y-muuttujien arvot.
- Voidaan käyttää esimerkiksi ohjelmien debuggauksessa, tai jos halutaan lopettaa ohjelman suoritus tietyn ehdon täyttyessä.

- Vastaavia proseduureja on for, if ja while rakenteiden lopettamiseen, mutta näitä käyttäen ohjelma suoritetaan muuten loppuun. Lisätietoja manuaalista.